Ciências ULisboa | Informática

di.ciencias.ulisboa.pt

/difcul

Segurança e Confiabilidade

2023/2024

Trabalho 1 – 2ª fase

1 Objetivos

A parte prática da disciplina de Segurança e Confiabilidade pretende familiarizar os alunos com alguns dos problemas envolvidos na programação de aplicações distribuídas seguras, nomeadamente a gestão de chaves criptográficas, a geração de sínteses seguras, cifras e assinaturas digitais, e a utilização de canais seguros à base do protocolo TLS. O primeiro trabalho a desenvolver na disciplina será realizado utilizando a linguagem de programação Java e a API de segurança do Java, e é composto por duas fases.

O trabalho desenvolvido da primeira fase envolveu a implementação de um sistema de IoT com arquitetura cliente-servidor mostrada na Fig.1. O programa **cliente**, nomeado `IoTDevice`, representa um dispositivo de sensorização. Este programa é responsável por enviar dados sensoriais para o servidor. Os dispositivos (*e.g.*, `luis@sc.pt:2`, `luis@sc.pt:5`) que executam o programa `IoTDevice` podem fazer parte de um ou mais domínios de trabalho (*e.g.*, `Room02`), e somente os utilizadores com acesso de leitura a um dado domínio (*e.g.*, `luis@sc.pt`, `alan@sc.pt`, no caso de `Room02`) podem aceder aos dados gerados por qualquer dispositivo dentro desse domínio.

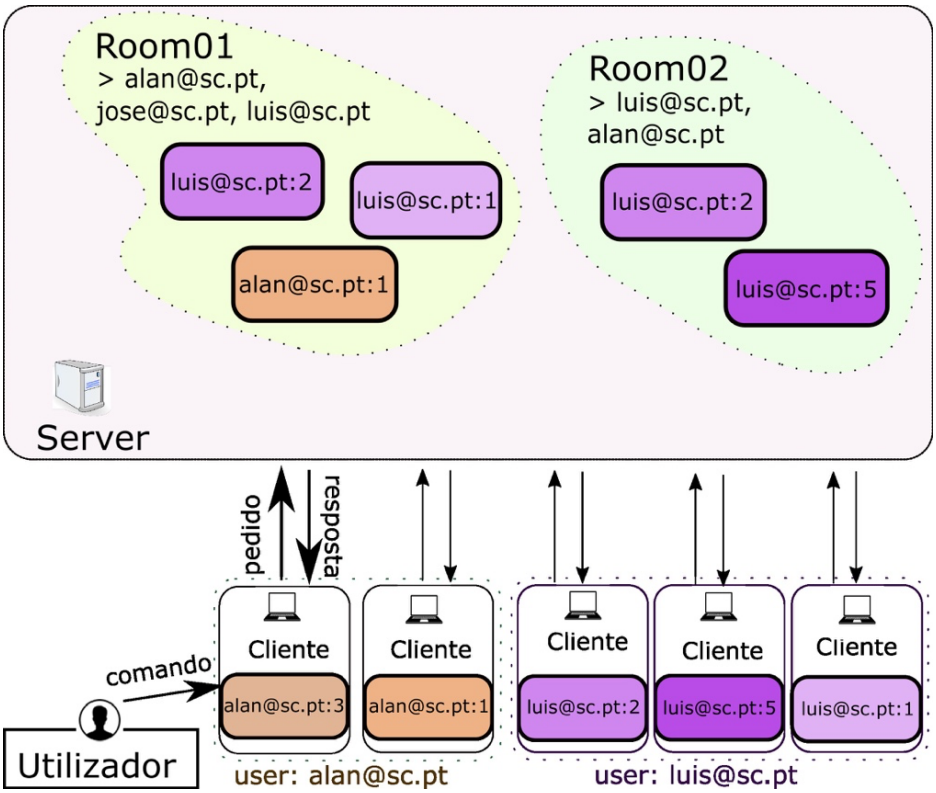

Fig.1. Diagrama da arquitetura do sistema IoT, com um Servidor e múltiplos clientes.

A aplicação `IoTDevice` tem duas grandes funcionalidades: 1) Enviar (dois tipos de) dados para o servidor: imagens e valores de temperatura; 2) Aceder a dados de temperatura e/ou imagens de um dado dispositivo. A aplicação **servidor,** nomeada `IoTServer`, é um programa que permite a ligação com vários clientes em simultâneo, mantém informação sobre os dispositivos, domínios e utilizadores registados, autentica e identifica utilizadores, e vai colecionando e partilhando informação já recebida dos vários clientes, de forma organizada e persistente.

A primeira fase do projeto teve como objetivo fundamental a construção de uma aplicação distribuída, tendo sido concretizadas as funcionalidades oferecidas pelo serviço. Nesta segunda fase do projeto serão considerados requisitos de segurança, para que as interações e o sistema no seu todo sejam seguros. As várias funcionalidades definidas na primeira fase serão mantidas sem qualquer alteração, embora a sua concretização tenha em alguns casos de ser adaptada para satisfazer os requisitos de segurança.

2 Modelo de sistema e definições preliminares

A arquitetura do sistema segue o que foi definido na primeira fase do projeto. Contudo, agora **todas as comunicações serão feitas através de *sockets* seguros TLS com autenticação unilateral**.

As chaves privadas estarão armazenadas em ***keystores*** (uma por cada utilizador e uma para o servidor) protegidas por passwords. Para além disso, e dado que todos os utilizadores terão necessidade de usar as chaves públicas uns dos outros e do servidor, os certificados de chave pública (auto-assinados) do servidor e dos clientes devem ser adicionados a uma ***truststore*** a ser usada por todos os clientes. Todos os pares de chaves serão gerados com o algoritmo RSA de 2048 bits.

Para maximizar ainda mais a confiança no ambiente de execução, **o servidor tem de cifrar o ficheiro de utilizadores.** Isto garante que ninguém, além do servidor, consegue ler este ficheiro. A chave a ser usada na cifra do ficheiro deve ser baseada numa password apresentada na inicialização do servidor, usando, portanto, o algoritmo PBE (Password Based Encryption) com AES de 128 bits. Quanto aos restantes ficheiros mantidos pelo servidor (ficheiros contendo informações dos domínios de trabalho e dados para atestação remota da aplicação), será necessário assegurar que não foram corrompidos, **verificando sempre a sua integridade** antes de usar a informação neles contida.

Será ainda necessário **criar um mecanismo para atestação remota da aplicação cliente**. Este mecanismo deve ser seguro em relação a ataques do tipo *replay*.

Finalmente, será necessário **garantir confidencialidade end-to-end** para os dados (temperatura e imagem) partilhados por um dispositivo no âmbito de um determinado domínio. Isto significa que os dados partilhados poderão ser decifrados todos os usuários pertencentes ao mesmo domínio em que o dispositivo está registado. Usuários não pertencentes a este domínio não poderão decifrar tais dados, assim como o servidor também não poderá.

3 Utilização do Sistema Seguro

A execução das aplicações servidor e cliente deve ser feita da seguinte forma:

1. `IoTServer <port> <password-cifra> <keystore> <password-keystore> <2FA-APIKey>`
    - `<port>` identifica o porto (TCP) para aceitar ligações de clientes. Por omissão o servidor deve usar o porto 12345;
    - `<password-cifra>` é a password a ser usada para gerar a chave simétrica que cifra os

ficheiros da aplicação;

- `<keystore>` que contém o par de chaves do servidor;
- `<password-keystore>` é a password da *keystore*;
- `<2FA-APIKey>` é uma chave dada a cada grupo de alunos para o processo de autenticação de dois fatores definido na Secção 4.2.

2. `IoTDevice <serverAddress> <truststore> <keystore> <password-keystore> <dev-id> <user-id>`

    - `<serverAddress>` identifica o servidor. O formato de serverAddress é o seguinte: `<IP/hostname>[:Port]`. O endereço IP ou *hostname* do servidor são obrigatórios e o porto é opcional. Por omissão, o cliente deve ligar-se ao porto 12345 do servidor;
    - `<truststore>` que contém os certificados de chave pública do servidor e dos utilizadores;
    - `<keystore>` que contém o par de chaves do user-id;
    - `<password-keystore>` é a password da *keystore*;
    - `<dev-id>` - número inteiro que identifica o dispositivo.
    - `<user-id>` – string que identifica o (**endereço de email do**) utilizador local.

4 Adicionar Segurança ao Sistema

4.1 Canais seguros TLS para comunicação segura e autenticação de servidores

Dado que na primeira fase do trabalho a comunicação entre cliente e servidor era feita de forma insegura (canais em claro e sem autenticação do servidor), nesta segunda fase será necessário resolver este problema de segurança. Em concreto, será necessário garantir a **autenticidade do servidor** (um atacante não deve ser capaz de fingir ser o servidor e assim obter os dados do utilizador) e a **confidencialidade** da comunicação entre cliente e servidor (um atacante não deve ser capaz de escutar a comunicação). Para este efeito, devem-se usar **canais seguros** (protocolo TLS/SSL) e a verificação da identidade do servidor à base de criptografia assimétrica (fornecida pelo protocolo TLS). Nesta fase do trabalho é então necessário:

- Utilizar ligações TLS: deve-se substituir a ligação TCP por uma ligação TLS/SSL, uma vez que o protocolo TLS vai verificar a autenticidade do servidor e garantir a integridade e confidencialidade de toda a comunicação.
- Configurar as chaves necessárias: a utilização do protocolo TLS exige configurar as chaves tanto no cliente (*truststore* com o certificado auto-assinado do servidor) como no servidor (*keystore* com a sua chave privada e certificado da sua chave pública).

4.2 Autenticação de dois fatores para utilizadores

Diferentemente da primeira fase do projeto, **não vamos utilizar autenticação via password de utilizador**, mas sim um mecanismo de autenticação de dois fatores. O primeiro fator, especificado na Secção 4.2.1, corresponde a uma autenticação baseada em criptografia assimétrica. O segundo fator, especificado na Secção 4.2.2, requer um código enviado por e-mail ao utilizador a cada tentativa de autenticação. Deste modo, um usuário só é autenticado se obtiver sucesso nos dois fatores.

4.2.1 Autenticação baseada em criptografia assimétrica

A autenticação baseada em criptografia assimétrica será feita em dois passos (após a ligação TLS ser estabelecida):

1) Cliente envia *user-id* ao servidor pedindo para se autenticar. O servidor responde com um *nonce* aleatório de 8 bytes (por exemplo, um *long*) e, caso o utilizador não esteja registado, uma *flag* a identificar que o utilizador é desconhecido.
2) Duas possibilidades:
   a) Se *user-id* ainda não tiver sido registado (i.e., é desconhecido), o cliente efetua o registo do utilizador enviando ao servidor o *nonce* recebido, a assinatura deste gerada com a sua chave privada, e o certificado com a chave pública correspondente. O servidor verifica se o *nonce* recebido foi o gerado por ele no passo 1) e se a assinatura pode ser corretamente verificada com a chave pública enviada (provando assim que o cliente tem acesso à chave privada daquele utilizador, i.e., que é quem diz ser). Se esses testes forem bem-sucedidos, o servidor completa o registo, armazenando o par <*user-id*>:<chave pública> no ficheiro *users.txt* (que agora deve ser cifrado pelo servidor), onde o parâmetro <chave pública> é o nome do ficheiro que contém o certificado do utilizador. Após a conclusão do registo, o servidor envia ao cliente uma mensagem a informar que o utilizador está registado e o fator de autenticação baseado em criptografia assimétrica foi bem-sucedido. Caso o *nonce* ou a assinatura sejam inválidos, é devolvida uma mensagem de erro ao cliente informando que o registo e a autenticação não foram bem-sucedidos.
   b) Se *user-id* estiver registado no servidor, o cliente assina o *nonce* recebido com a sua chave privada, e envia esta assinatura ao servidor. O fator de autenticação baseado em criptografia assimétrica é bem-sucedido apenas se o servidor verificar a assinatura do *nonce* (que deve ser o mesmo gerado e enviado no passo 1) usando a chave pública associada ao *user-id*. Caso não se verifique a assinatura, é enviada uma mensagem de erro ao cliente informando que a autenticação não foi bem-sucedida.

De notar que neste nesta 2ª fase, o *dev-id* já não é verificado durante o processo de autenticação, mas será verificado durante o processo de atestação remota (Secção 4.3).

4.2.2 Confirmação baseada em e-mail enviado ao utilizador

Se a autenticação baseada em criptografia assimétrica especificada na Secção 4.2.1 for bem sucedida, executa-se então o segundo mecanismo da autenticação de dois fatores. Este mecanismo se baseia na verificação de um código C_{2FA} enviado por email ao utilizador, conforme os passos abaixo:

1) O `IoTServer` gera o código C_{2FA}, que corresponde a um número aleatório de cinco dígitos (entre 00000 e 99999).
2) Para enviar o C_{2FA} por e-mail ao utilizador, o `IoTServer` faz GET a uma API REST disponibilizada pelos docentes da disciplina utilizando a URL abaixo:

   `https://lmpinto.eu.pythonanywhere.com/2FA?e=`**`<user-id>`**`&c=`**`<C2FA>`**`&a=`**`<2FA-APIkey>`**

   onde **<user-id>** é o endereço de e-mail do utilizador que se está a autenticar, **<C_{2FA}>** é o código gerado no passo anterior, e **<2FA-APIkey>** é uma chave dada a cada grupo de alunos e que limita o acesso ao serviço de envio de e-mails a uma chamada a cada 5 segundos. De notar que o endereço de email tem de ser um endereço válido, por exemplo,

fcxxxxx@alunos.ciencias.ulisboa.pt, para que se consiga receber o código enviado pelo servidor).

3) Ao receber o código C_{2FA} por e-mail, o utilizador deve introduzi-lo no terminal da aplicação `IoTDevice` (pode ser explicitamente solicitado ao utilizador com uma *prompt* "Introduza o código enviado pelo servidor:");
4) O código C_{2FA} introduzido pelo utilizador é enviado pelo `IoTDevice` ao `IoTServer`;
5) O `IoTServer` compara o código C_{2FA} recebido com aquele que foi gerado no passo 1. Se os dois forem iguais, então o servidor considera o usuário autenticado e o notifica da autenticação bem-sucedida. Se os códigos forem diferentes, então o servidor considera a autenticação falhada e notifica o usuário sobre o insucesso. Neste caso, se o usuário quiser realizar uma nova tentativa de autenticação, deverá refazer todo o processo de autenticação de dois fatores.

4.3 Atestação Remota da Aplicação

O servidor deverá fazer a atestação remota da aplicação cliente. Na primeira fase, a atestação remota era feita apenas com base no tamanho do ficheiro executável `IoTDevice`. Nesta segunda fase, a atestação remota será feita com base no *hash* SHA256 do ficheiro executável `IoTDevice`.

Para que este mecanismo de atestação remota seja seguro a ataques do tipo *replay*, é necessário implementar os seguintes passos:

1) Após o utilizador concluir a sua autenticação, a aplicação `IoTDevice` envia ao servidor o `<dev-id>`. Se o `<dev-id>` for inválido, ou seja, se o dispositivo com identificação (`<user-id>`,`<dev-id>`) já estiver ativo, o servidor deve responder com NOK-DEVID, a ligação é terminada, e o cliente termina assinalando o erro respetivo. Se o `<dev-id>` enviado ao servidor for válido, o servidor enviará ao cliente a mensagem OK-DEVID agora acompanhada por um *nonce* aleatório de 8 bytes (por exemplo, um *long*).
2) Ao receber o *nonce*, o cliente o concatenará com o ficheiro executável `IoTDevice`, calculará o *hash* SHA256 desta concatenação e enviará o *hash* calculado para o servidor.
3) Ao receber o *hash* calculado pelo cliente, o servidor irá compará-lo com o *hash* calculado localmente com base na concatenação entre o mesmo *nonce* e uma cópia de referência da aplicação `IoTDevice` guardada no servidor. Dois cenários podem ocorrer:
    a. O servidor não valida o programa cliente – o cliente recebe a mensagem do tipo NOK-TESTED, a ligação é terminada, e o cliente termina assinalando o erro respetivo.
    b. O servidor valida o programa cliente – o cliente recebe a mensagem do tipo OK-TESTED, e o processo segue em frente.

Note que, no servidor, o ficheiro de texto que antes guardava o nome e o tamanho da aplicação, agora não mais terá a informação de tamanho. No lugar desta, o ficheiro terá a diretoria e nome da cópia de referência da aplicação `IoTDevice` guardada localmente.

4.4 Confidencialidade fim-a-fim para as mensagens

Na 1ª fase do projeto os dados enviados por um dispositivo podiam ser lidos pelo servidor e eram guardados por este em claro num ficheiro. Pretende-se agora nesta fase que seja assegurada confidencialidade fim-a-fim, ou seja, que apenas os utilizadores pertencentes ao(s) domínio(s) aos quais o dispositivo pertence possam ler os dados. Para tal, os dados enviados ao servidor serão cifrados com criptografia simétrica (PBE com AES de 128 bits) utilizando uma **chave de domínio**. Esta chave, criada pelo Owner do domínio, será partilhada apenas com os demais utilizadores do domínio

por meio de criptografia assimétrica. Para isso, os seguintes pontos devem ser observados:

1) A chave de domínio (simétrica) é criada pelo Owner do domínio com base numa password e nos demais parâmetros requeridos para a geração da chave (*i.e.*, *salt* e número de iterações). Note que se for necessário produzir novamente a mesma chave de domínio, é preciso usar a mesma password e os mesmos valores para os referidos parâmetros. Por este motivo, os valores destes parâmetros devem ser guardados;
2) Toda vez que um Owner adicionar um utilizador a um domínio, este deverá enviar ao servidor a chave de domínio cifrada com a chave pública do utilizador. O comando de adição de utilizador a um domínio passa agora a ser: `ADD <user1> <dm> <password-domínio>`, em que `<password-domínio>` é usada para produzir a chave de domínio, juntamente com os parâmetros usados aquando da criação inicial da chave.
3) Toda vez que um utilizador pedir dados (temperaturas de um domínio ou uma imagem de um <user-id>:<dev_id>), o servidor deverá verificar se o utilizador tem permissão de acesso com base nos domínios em que este se encontra registado. Se o utilizador tiver permissão de acesso, o servidor enviará os dados solicitados juntamente com a chave do domínio respetivo (cifrada com a chave pública do utilizador). Ao receber a chave de domínio cifrada, o `IoTDevice` a decifrará usando chave privada do utilizador. Em seguida, o `IoTDevice` decifrará os dados usando a chave de domínio. Se o utilizador não tiver permissão de acesso, o servidor enviará ao cliente `IoTDevice` uma mensagem NOPERM.
4) Toda vez que um dispositivo for enviar um dado (temperatura ou imagem) ao servidor, o cliente `IoTDevice` deverá enviar ao `IoTServer` cópias do dado cifradas com as chaves dos diferentes domínios a que o dispositivo pertence.

5 Funcionalidades

Deve ser acrescentada a seguinte nova funcionalidade ao cliente `IoTDevice`:

- `MYDOMAINS` – imprime a lista de domínios que o dispositivo pertence.

Note que, para implementar esta funcionalidade, os grupos podem usar chamadas eventualmente criadas no ponto 4) da secção 4.4 para obter a lista de domínios a que o dispositivo pertence.

6 Entrega

6.1 Trabalho

Dia **26 de abril**, até às 23:59 horas. O código desta segunda (e última) fase do trabalho deve ser entregue de acordo com as seguintes regras:

- A segunda fase do projeto deve ser realizada mantendo os grupos da primeira fase.
- Para entregar o trabalho, é necessário criar um **ficheiro zip** com o nome **SegC-grupoXX-proj1-fase2.zip**, onde **XX** é o número do grupo, contendo:
  - ✓ o código do trabalho;
  - ✓ conjunto de chaves, *keystores*, certificados (ficheiros de formato **cert**) e *truststore*;
  - ✓ os ficheiros jar (cliente e servidor) para execução do projeto;
  - ✓ um readme (txt) indicando **como compilar** e **como executar** o projeto, indicando também as limitações do trabalho.
- O ficheiro zip é submetido através da atividade disponibilizada para esse efeito na página da disciplina no Moodle. Apenas um dos elementos do grupo deve submeter. Se existirem várias submissões do mesmo grupo, será considerada a mais recente.

- Não serão aceites trabalhos enviados por email. Se não se verificar algum destes requisitos o trabalho é considerado não entregue.

6.2 Autoavaliação de contribuições

Dia **28 de abril**, até às 23:59 horas. Cada aluno preenche no Moodle um formulário de autoavaliação das contribuições individuais de cada elemento do grupo na 2º fase do projeto. Por exemplo, se todos os elementos colaboraram de forma idêntica, bastará que todos indiquem que cada um contribuiu 33%. Aplicam-se as seguintes regras e penalizações:

- Alunos que não preencham o formulário sofrem uma penalização na nota de 10%.
- Caso existam assimetrias entre as contribuições de cada elemento do grupo ou entre as autoavaliações de cada elemento do grupo, estas serão analisadas durante a discussão do trabalho e poderão levar à atribuição de notas individuais diferentes para cada elemento do grupo.